QUICK START GUIDE

Xilinx SystemC Qemu Co-Simulation

Revision 1.0

3/17/22

Wojciech Zebrowski updated

Radoslaw Wrobel initiation

Abstract

This is a dedicated solution for Xilinx Zynq devices Co-Simulation with Riviera-PRO. It connects Xilinx QEMU via Remote Port to Riviera-PRO using Xilinx SystemC TLM library. This model can be directly simulated in Aldec Riviera-PRO which makes co-verification even simplier.

Meta Keywords Qemu Co-Simulation

Related Products Riviera-Pro simulator

Related Methodologies ASIC Prototyping

Related Markets Military, Aerospace, Avionics, Medical, Nuclear, Transportation, Telecommunications, Radar, High Frequency Trading, Gaming, Embedded, Automotive, IoT, UAV, High Performance Computing

Table of Contents

Resources

This solution is based on the blog article and some files created by Rick Wertenbroek (refer to: *https://blog.reds.ch/?p=1180*). A detailed technical description can be found on the Wiki Xilinx website: *https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/862421112/Co-simulation*.

Limitations

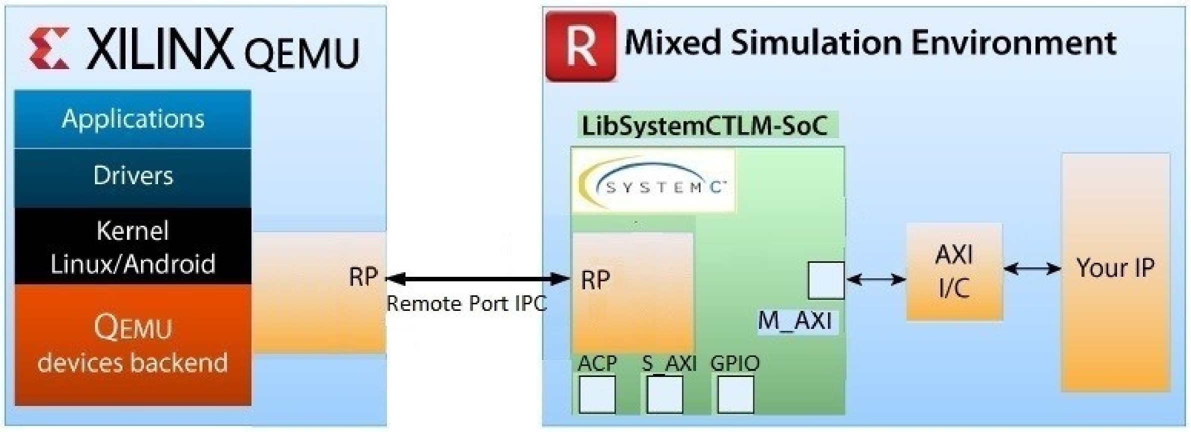
- Aldec Riviera-Pro 2021.10
- Xilinx Vivado 2021.1
- Xilinx Petalinux 2021.1
- Checked on CentOS and Ubuntu systems

Introduction

Today's SoC FPGAs present new verification challenges for system, software and hardware engineers. Common issues related to HW/SW integration continue to increase, and yet they are only typically found in the testbed with the SoC FPGA running. Finding the issues in the testbed is often too late and can cause project delays.

The situation on FPGA chips market is unpredictable. The emulation tools is good way to solve time to market delay and to bring tested and ready to use system out.

Aldec provides a HW/SW Co-Simulation interface between Riviera-PRO and Xilinx Quick Emulator QEMU. System integration and Co-Simulation of HDL code with software applications/drivers executing in QEMU is now simplified with full compilation SystemC to library (LibSysytemCTLM-SoC). It is executed under Riviera-Pro simulator. The LibSysytemCTLM-SoC shares Remote Port to connect Riviera-PRO and QEMU, and converts SystemC TLM transactions to AXI and vice versa providing a fast interface for Co-



Simulation.

Figure 1: HW/SW Co-Simulation Environment with Riviera-PRO and QEMU

Figure 1 presents the block diagram of HW/SW Co-Simulation Environment with Riviera-PRO and QEMU.

This document will focus on executing an example project in Co-Simulation and describing the available scripts to automate the building process.

Preparing Co-Simulation sources

This chapter describes a process of preparing Co-Simulation environment. The first step requires a Vivado Zynq SoC project from which the simulation model is generated. The obtained simulation model is then connected to the TLM sockets and the Zynq processing system is emulated on the Qemu platform. Communication between the SW and HW layers takes place via the Remote Port.

To facilitate the Co-Simulation process, Aldec provides a repository with scripts to building and running project files. The repository is based on the example Vivado project described in the next chapter. Nothing prevents from replacing this project with others and taking full advantage of the possibility to Co-Simulate own solution.

To download the repository, run the command:

```
git clone http://ADD_PROPER_PATH/TySOM-QEMU.git
```

Preparation of HW project in Vivado

The easiest way to start creating a Vivado project that will be Co-Simulated is to use the Aldec Board files. Using Vivado board definitions allows to faster creating systems. After selecting a specific board, the project is configured with the predefined interface for that board and the Vivado design tools provide designer assistance to IP customization and creating IP integrator designs. More details how to build a Vivado design using board definition can be found below:

https://github.com/aldec/TySOM/blob/master/Vivado-board_files/How_to_build_an_FPGA_design_for_Aldec_TySOM_boards_using_board_definition_in_Xilinx_Vivado.pdf

An example Vivado project to demonstrate the possibilities of Qemu SystemC Co-Simulation was prepared as follows.

1. Download Aldec TySOM BSP from Github.
   ```
   git clone https://github.com/aldec/TySOM
   ```
2. Source Vivado setting and run Vivado.
3. Add Aldec Board files to Xilinx Vivado using TCL command:
   ```
   set_param board.repoPaths [list <path1> <path2> ]
   ```
4. Create the design for TySOM-1-7Z030
5. Create block design.
6. Add Zynq-7 Processing System IP core.
7. Go to the Board tab and add pre-configured for TySOM-1 PL user LEDs and PL user switches.
8. Run all the connection automation.
9. The final block design should be as on the Figure 2.
10. The axi_gpio_0 (user switches) is on 0x41200000 address. The axi_gpio_1 is on 0x41210000 address. Save and validate the design.
11. Create HDL Wrapper and generate bitstream.
12. Export Hardware including the bitstream.

13. As a result, an .xsa file should be created, which will be used in the next step to prepare the Petalinux project.

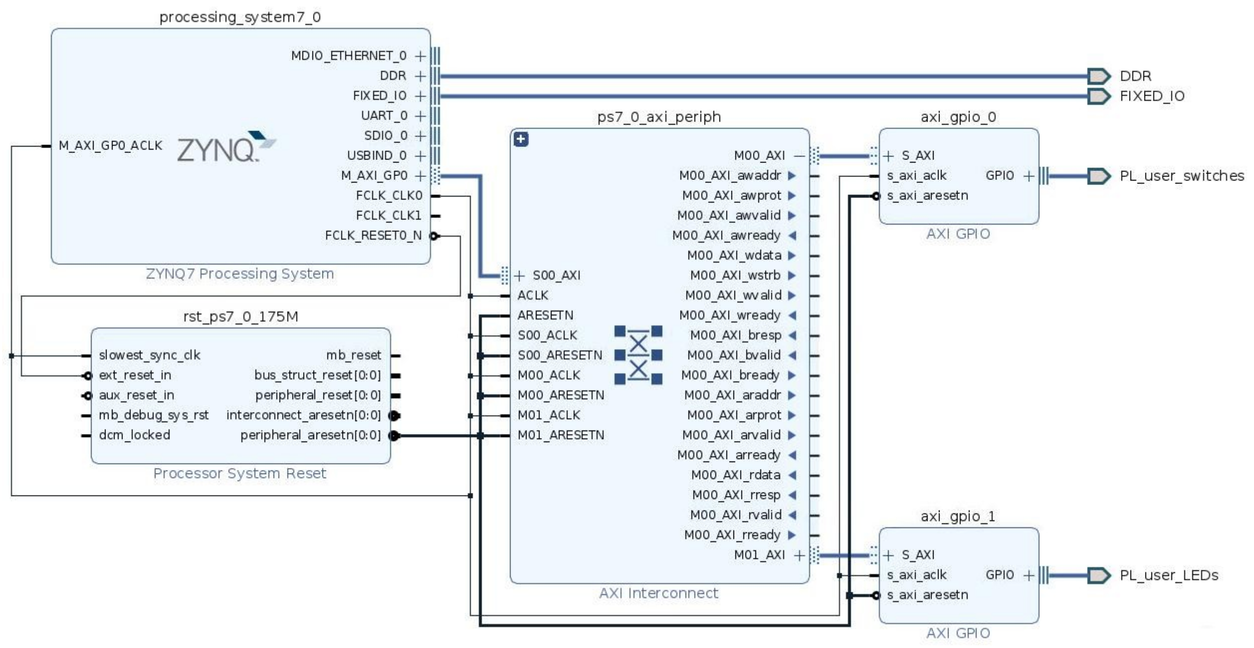


Figure 2: View of the example Vivado block design

Preparing design for Co-Simulation

To prepare the Co-Simulation files is needed a Xilinx Support Archive (XSA). This file is prepared and available in the repository in the example_hw folder. To prepare a Co-Simulation based on own project, just replace the default xsa file with another. The generation process will start using the xsa file from the example_hw folder.

Note: Before starting the scripts, set the correct paths to the tools. They are located in the config.sh file

```
export RIVIERA_PATH=<path_to_Aldec_Riviera-PRO-2021.10-x64>
export PETALINUX_PATH=<path_to_Xilinx_petalinux-v2021.1>
export VIVADO_PATH=<path_to_Vivado_2021.1>

# Path to Qemu Socket
RPORT_PATH=`pwd`/${project_name}/qemu_cosim/qemu-rport-_cosim@0
export RPORT_PATH_SC=${RPORT_PATH}
export RPORT_PATH_HDL=unix:${RPORT_PATH}
```

The process of generation sources is started by running the command below.

```
$ ./prepare_files.sh
```

Sample result from running script:

```
PetaLinux environment set to '/edatools/Xilinx/Linux/petalinux-v2021.1'
INFO: Checking free disk space
INFO: Checking installed tools
INFO: Checking installed development libraries
INFO: Checking network and other services
```

```
INFO: Create project: tysom1_petalinux
INFO: New project successfully created in
/home/systemc_qemu_cosim/tysom1_petalinux
[INFO] Sourcing buildtools
INFO: Getting hardware description...
INFO: Renaming design_1_wrapper.xsa to system.xsa
[INFO] Generating Kconfig for project
[INFO] Silentconfig project
[INFO] Extracting yocto SDK to components/yocto. This may take time!
[INFO] Sourcing build environment
[INFO] Generating kconfig for Rootfs
[INFO] Silentconfig rootfs
[INFO] Generating plnxtool conf
[INFO] Adding user layers
[INFO] Generating workspace directory
[INFO] Sourcing buildtools
[INFO] Building project
[INFO] Sourcing build environment
[INFO] Generating workspace directory
INFO: bitbake petalinux-image-minimal
[INFO] Sourcing buildtools
[INFO] Building device-tree
[INFO] Sourcing build environment
[INFO] Generating workspace directory
INFO: bitbake virtual/dtb
NOTE: Started PRServer with DBfile:
[INFO] Successfully built device-tree
Could find Processing System in file
Genenerated the follwing files :
riviera//../src/design_1_processing_system7_0_0.vhd
riviera//zynq7_compile_cosim.do
riviera//all.do
```

Riviera-Pro library

The project require rebuild simulation library under Vivad with Riviera-Pro simulator. Please prepare required library base on description:

https://www.aldec.com/en/support/resources/documentation/articles/1710

```
set XILINX_LIB_PATH "$env(RIVIERA_PATH)/Xilinx_lib"
amap -link $XILINX_LIB_PATH
```

After successfully building library for Riviera-Pro simulator, direct path should be set here:

$(Project_dir)/riviera/sim_top_compile.do

```
set XILINX_LIB_PATH "$env(RIVIERA_PATH)/Xilinx_lib"
amap -link $XILINX_LIB_PATH
```

Running Co-Simulation

After successfully building the sources, the last step is to run the Co-Simulation process. The easiest way to do this is to use a run_example_cosim.sh script.

```
./ run_example_cosim.sh
```

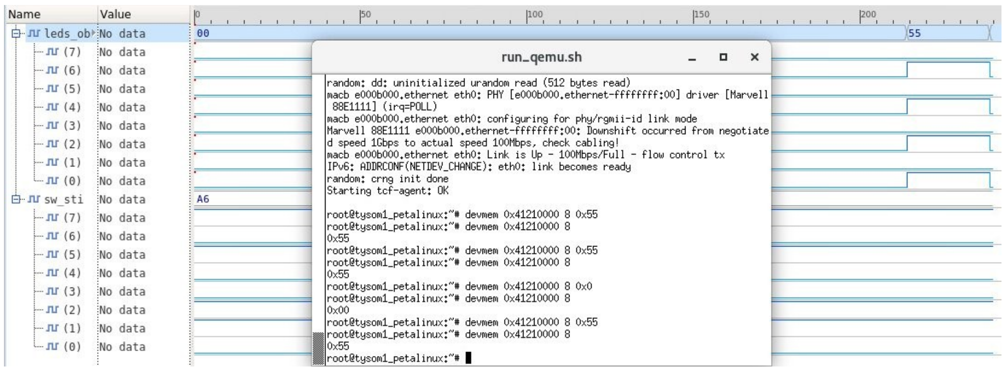
After booting the linux system, the LEDs can be tested with the devmem command:

```
root@tysom1_petalinux:# devmem 0x41210000 8 0x55
```

After that on the Riviera waveform should be a 0x55 value on the LEDs signals.

Above command should read back the value previously set on the LEDs.

```
root@tysom1_petalinux:# devmem 0x41210000 8
0x55
```



```
random: dd: uninitialized urandom read (512 bytes read)
macb e000b000.ethernet eth0: PHY [e000b000.ethernet-ffffffff:00] driver [Marvell
 88E1111] (irq=POLL)
macb e000b000.ethernet eth0: configuring for phy/rgmii-id link mode
Marvell 88E1111 e000b000.ethernet-ffffffff:00: Downshift occurred from negotiate
d speed 1Gbps to actual speed 100Mbps, check cabling!
macb e000b000.ethernet eth0: Link is Up - 100Mbps/Full - flow control tx
IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
random: crng init done
Starting tcf-agent: OK

root@tysom1_petalinux:~# devmem 0x41210000 8 0x55
root@tysom1_petalinux:~# devmem 0x41210000 8
0x55
root@tysom1_petalinux:~# devmem 0x41210000 8 0x55
root@tysom1_petalinux:~# devmem 0x41210000 8
0x55
root@tysom1_petalinux:~# devmem 0x41210000 8 0x0
root@tysom1_petalinux:~# devmem 0x41210000 8
0x00
root@tysom1_petalinux:~# devmem 0x41210000 8 0x55
root@tysom1_petalinux:~# devmem 0x41210000 8
0x55
root@tysom1_petalinux:~#
```

Figure 3: Example view of running Co-Simulation

Known issues

1. The Linux system will probably hand for quite a long time around below lines:

```
random: crng init done
Starting tcf-agent: OK
```

2. After Co-Simulation is finished, it is recommended to first turn off Riviera and then Qemu. The crash in the console may appear in the reverse order.